Detecting Inconsistencies in JavaScript MVC Applications

Frolin S. Ocariza, Jr.,
Karthik Pattabiraman and Ali Mesbah

University of British Columbia

Used by **90%** of all websites

Most popular language on GitHub (RedMonk survey)

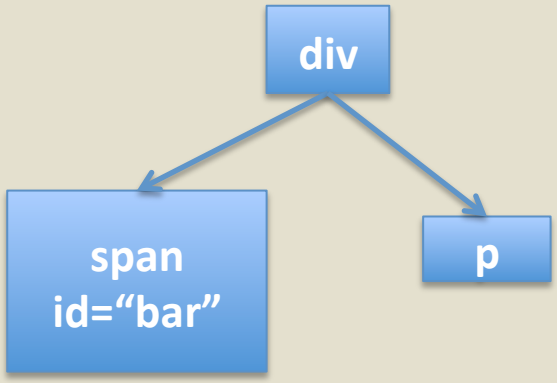
JavaScript

4 JavaScript error messages on average per website [ISSRE'11]

68% of JavaScript faults are ***DOM-related*** [ESEM'13]

DOM-Related Faults

DOM



JS CODE

```
var elem = getElementById("car")
elem.innerHTML = "Hello World!"
```

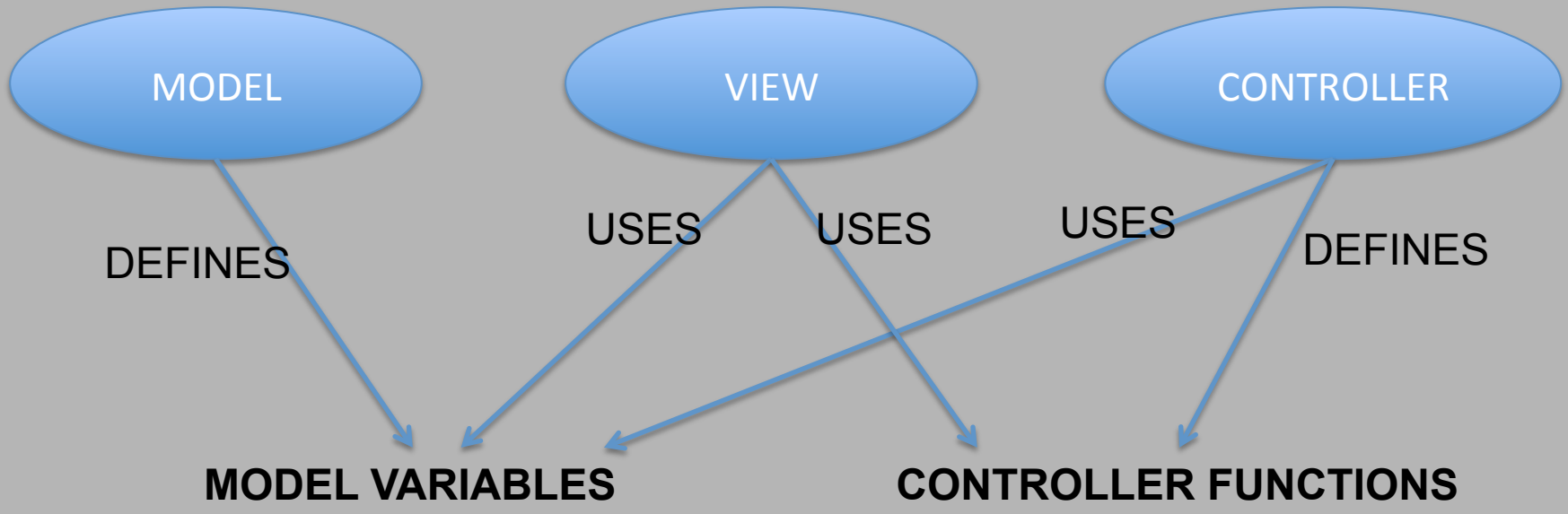
DOM API Method

Erroneous parameter

DOM-Related Faults are…

Prevalent **Impactful** **Non-Trivial to Fix**

MVC Frameworks to the Rescue!



*MVC Frameworks use **two-way data binding** ➔ No DOM method calls!*

MVC Frameworks to the Rescue!

MODEL

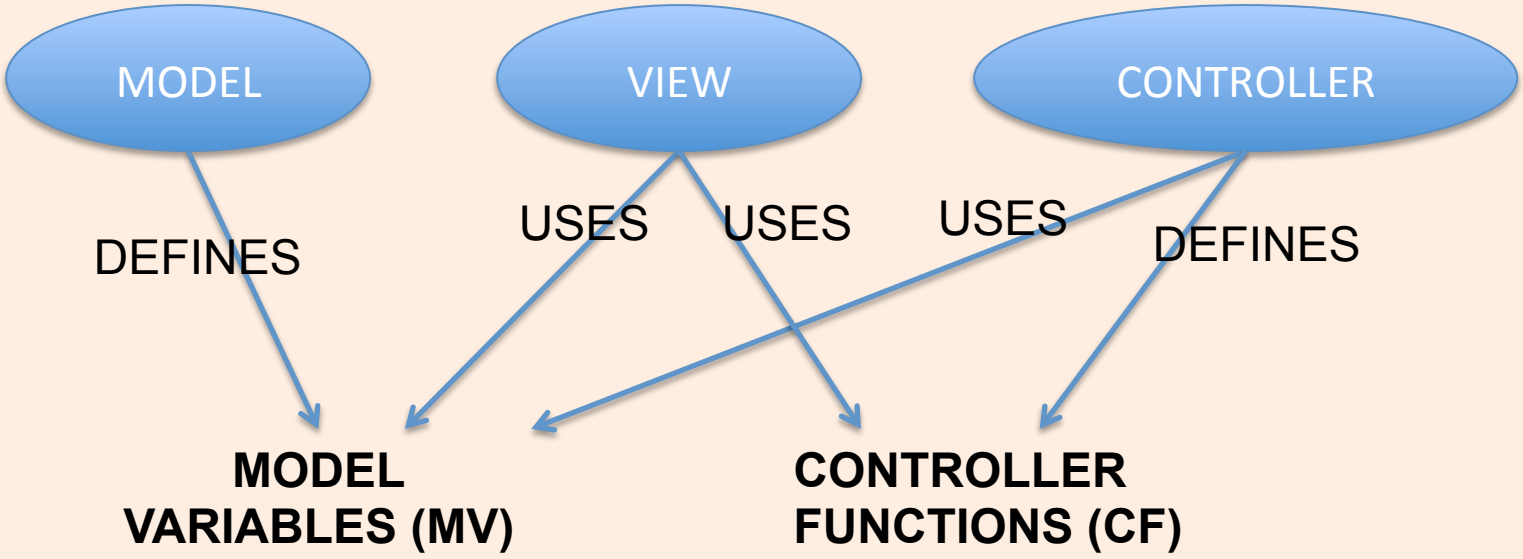
```
$scope.msg = "";
```

VIEW

```
<input type="text" ng-
   model="msg" />
<div ng-bind="msg"></div>
```

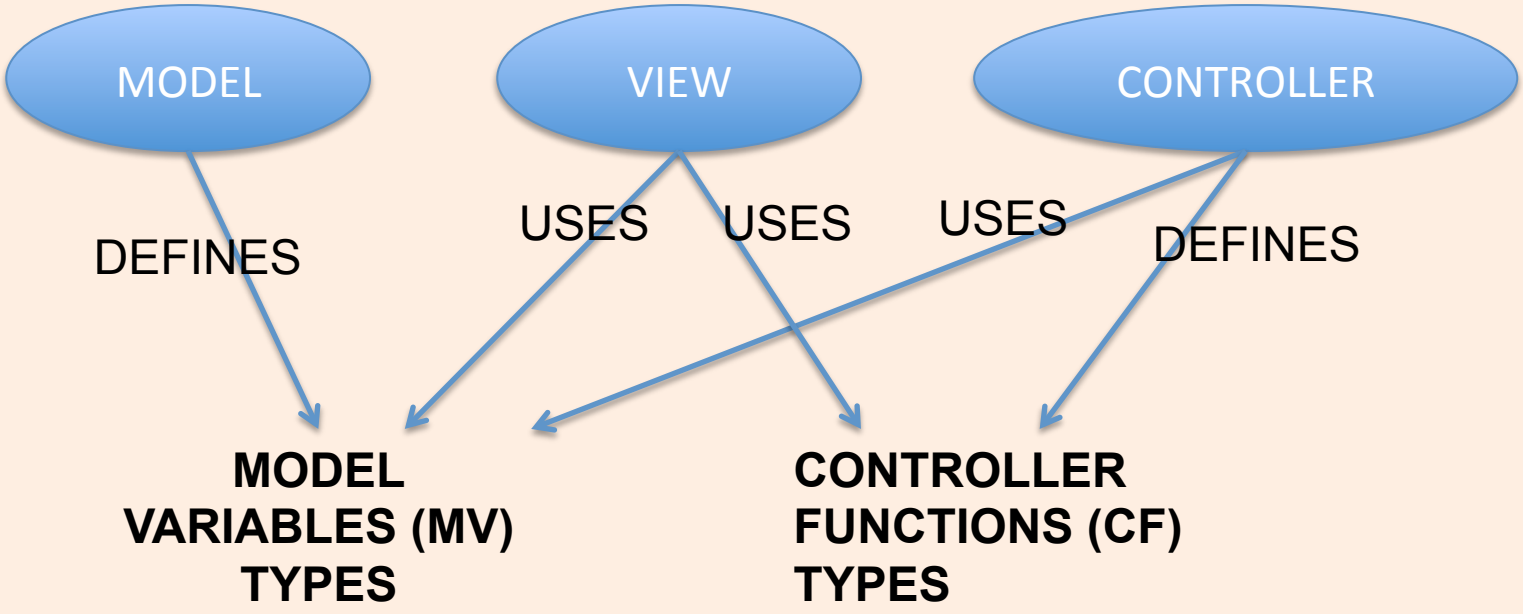
No DOM method calls ➔ **No DOM-Related Faults** in MVC Applications!

MVC Frameworks to the Rescue?



IDENTIFIER INCONSISTENCIES

MVC Frameworks to the Rescue?



IDENTIFIER INCONSISTENCIES

TYPE INCONSISTENCIES

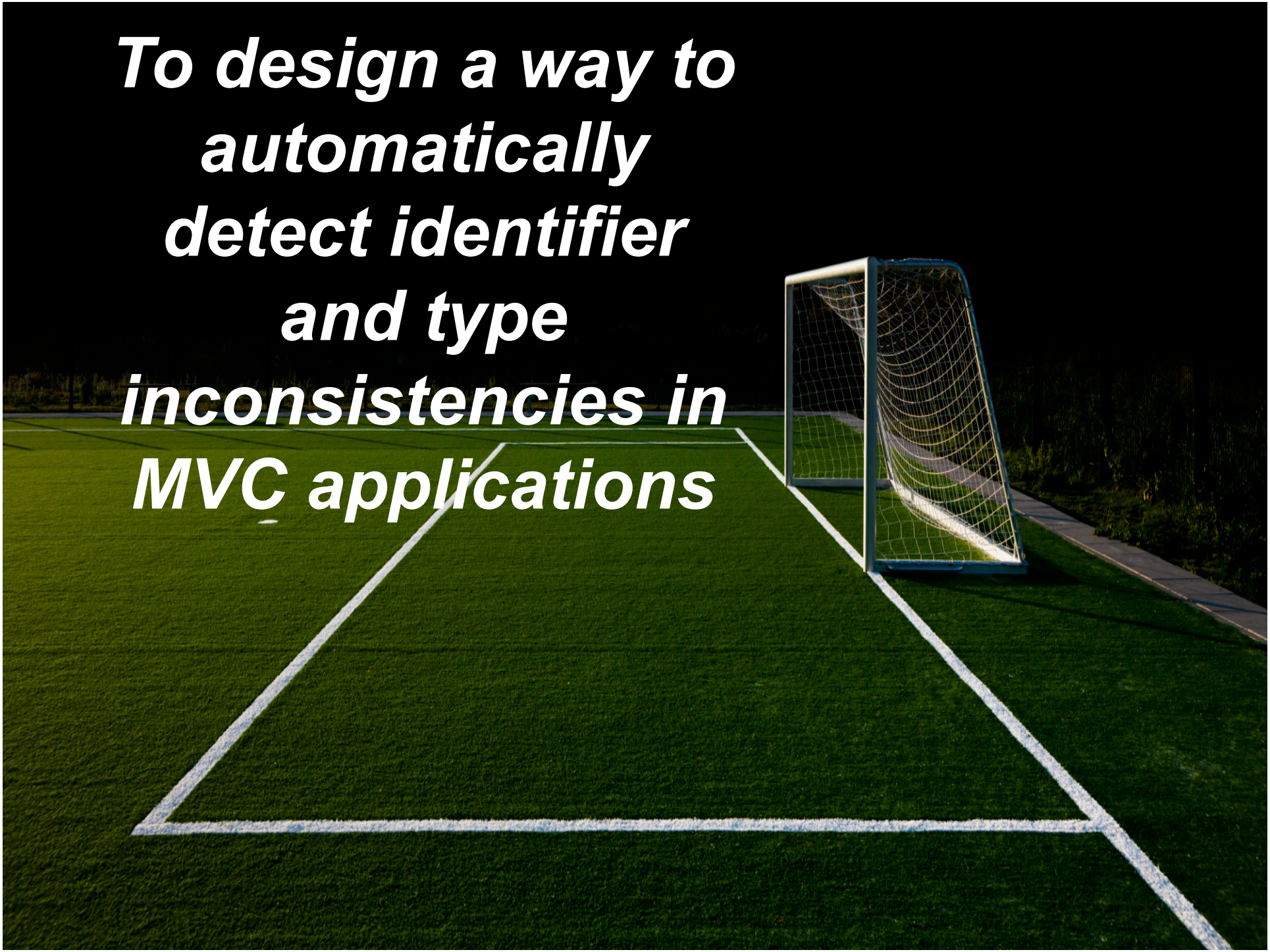
MVC Frameworks to the Rescue?

Identifier and type Inconsistencies happen in real life

- "What's wrong with AngularJS?" By B. Kinoshita
- "Why you should not use AngularJS" By E. Koshelko

Inconsistencies are hard to find

- ***e.g., No error messages thrown!***


To design a way to automatically detect identifier and type inconsistencies in MVC applications

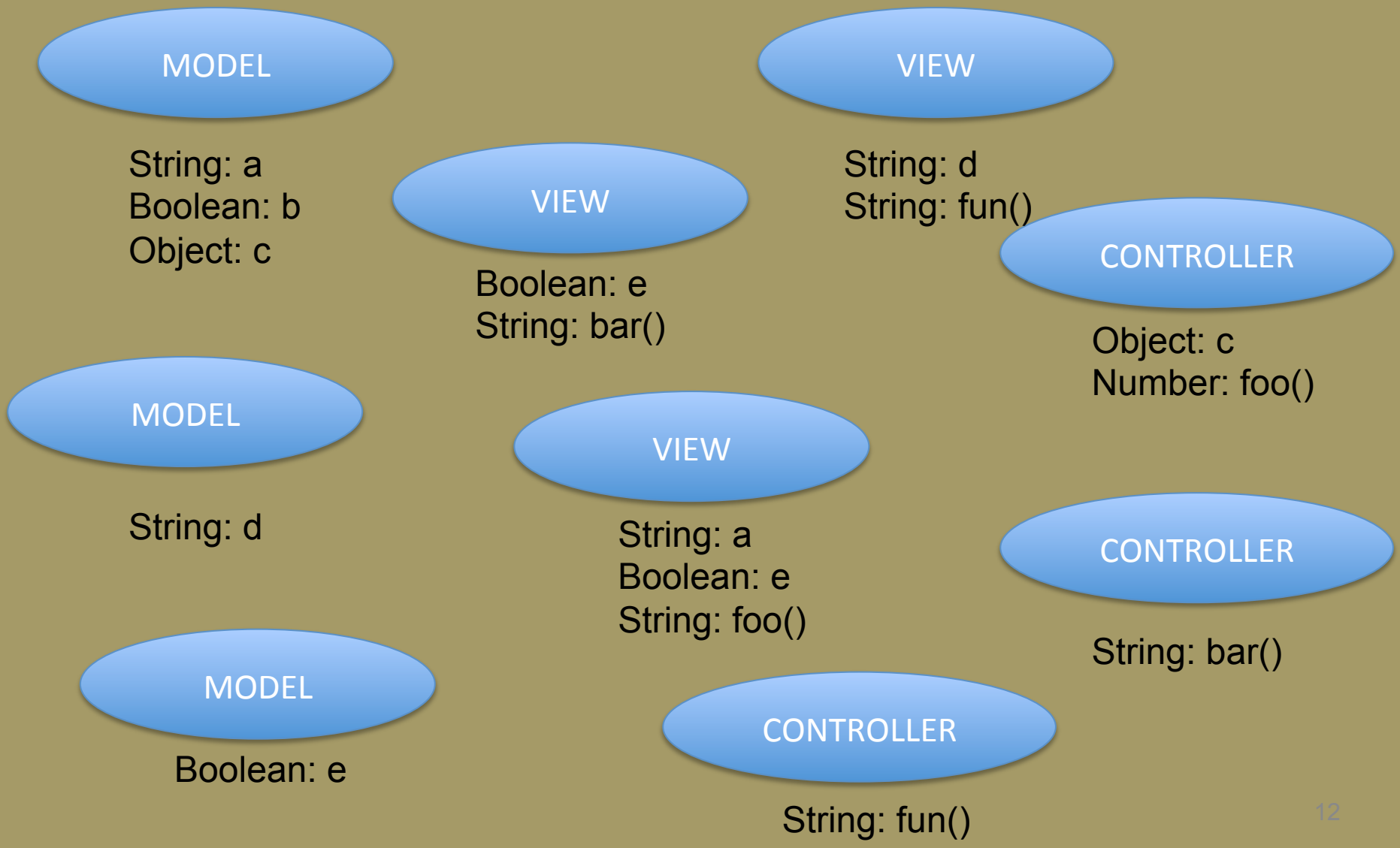
The **most popular JS MVC framework** in GitHub, StackOverfow, and YouTube

300% increase in AngularJS usage over the past year

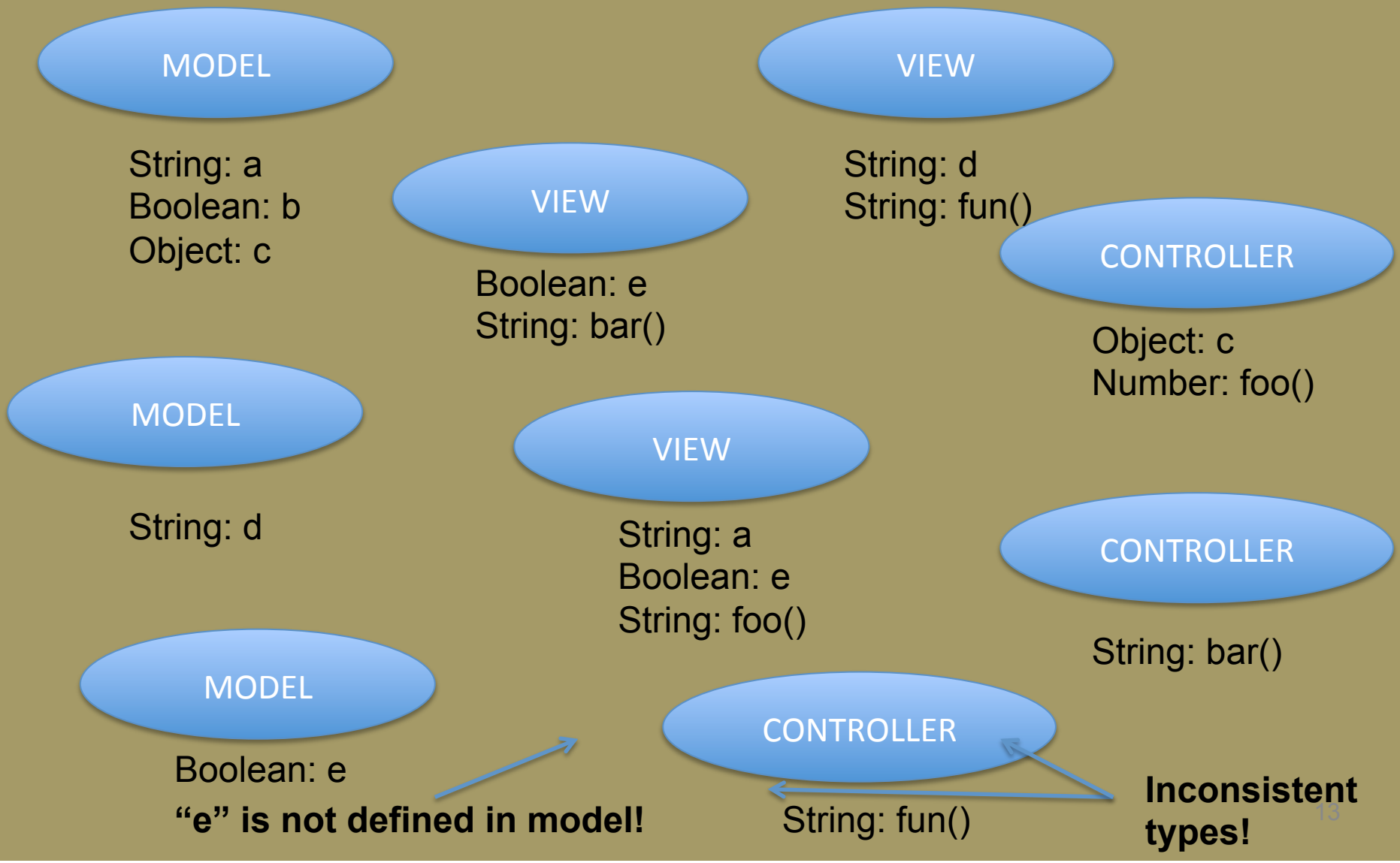
Our Approach

STATIC ANALYSIS

Our Approach



Our Approach



Challenges

Nested Objects

```
$scope.obj = {
    prop1: 2,
    prop2: {
        subprop1: “hello”,
        subprop2: true
    }
    prop3: “world”
};
```

`obj` is a model variable, but so is `obj.prop1`, `obj.prop2.subprop1`, etc.

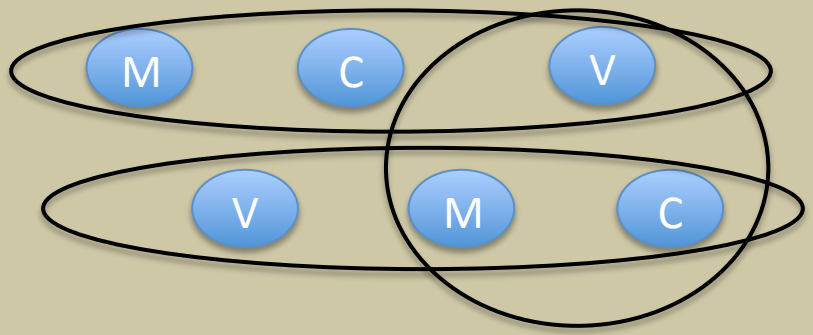
Aliasing

```
$scope.arr = […]
```

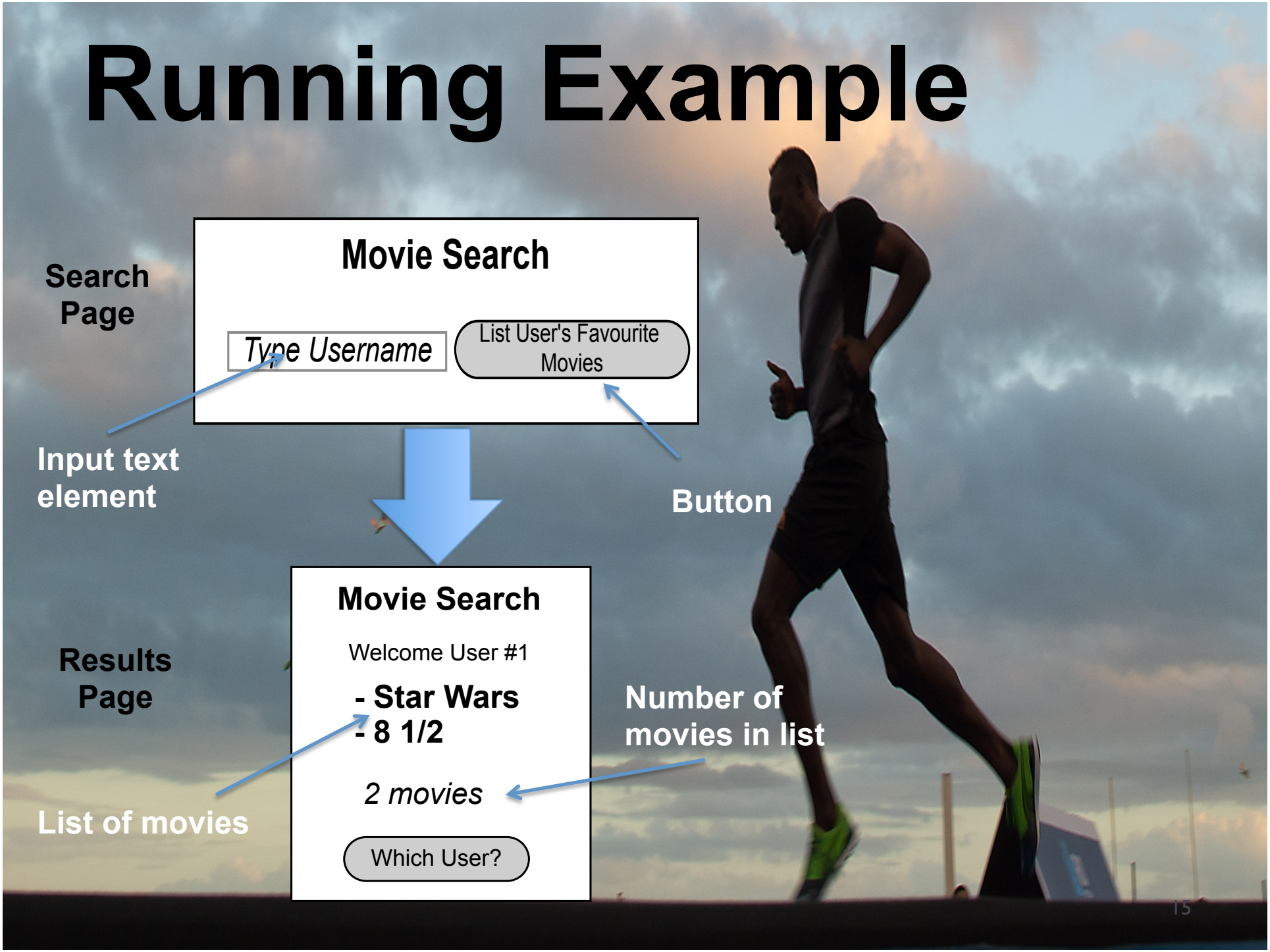
```
<li ng-repeat=“temp in arr”>
    {{temp.val}}
</li>
```

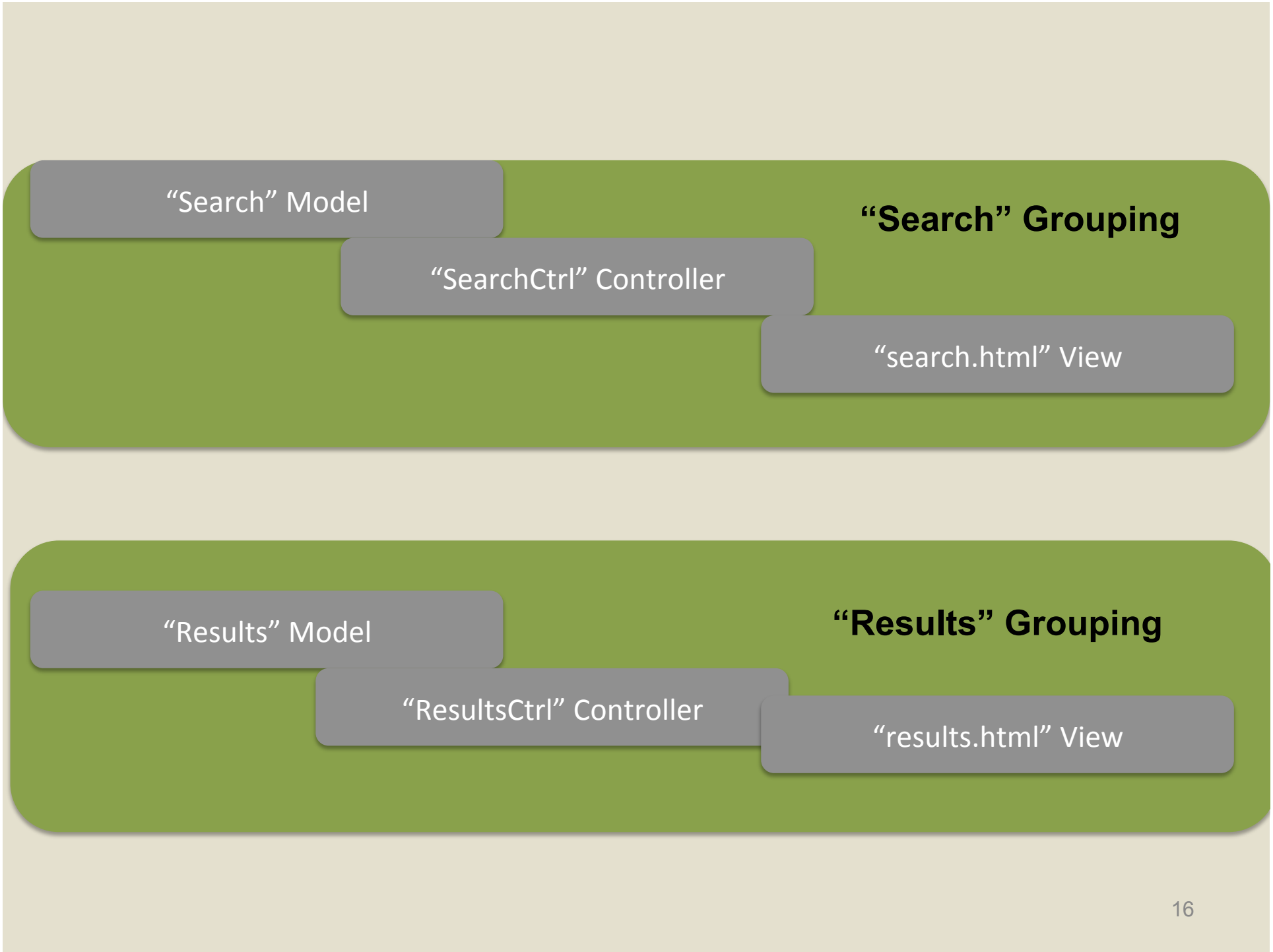
`temp` is an alias!

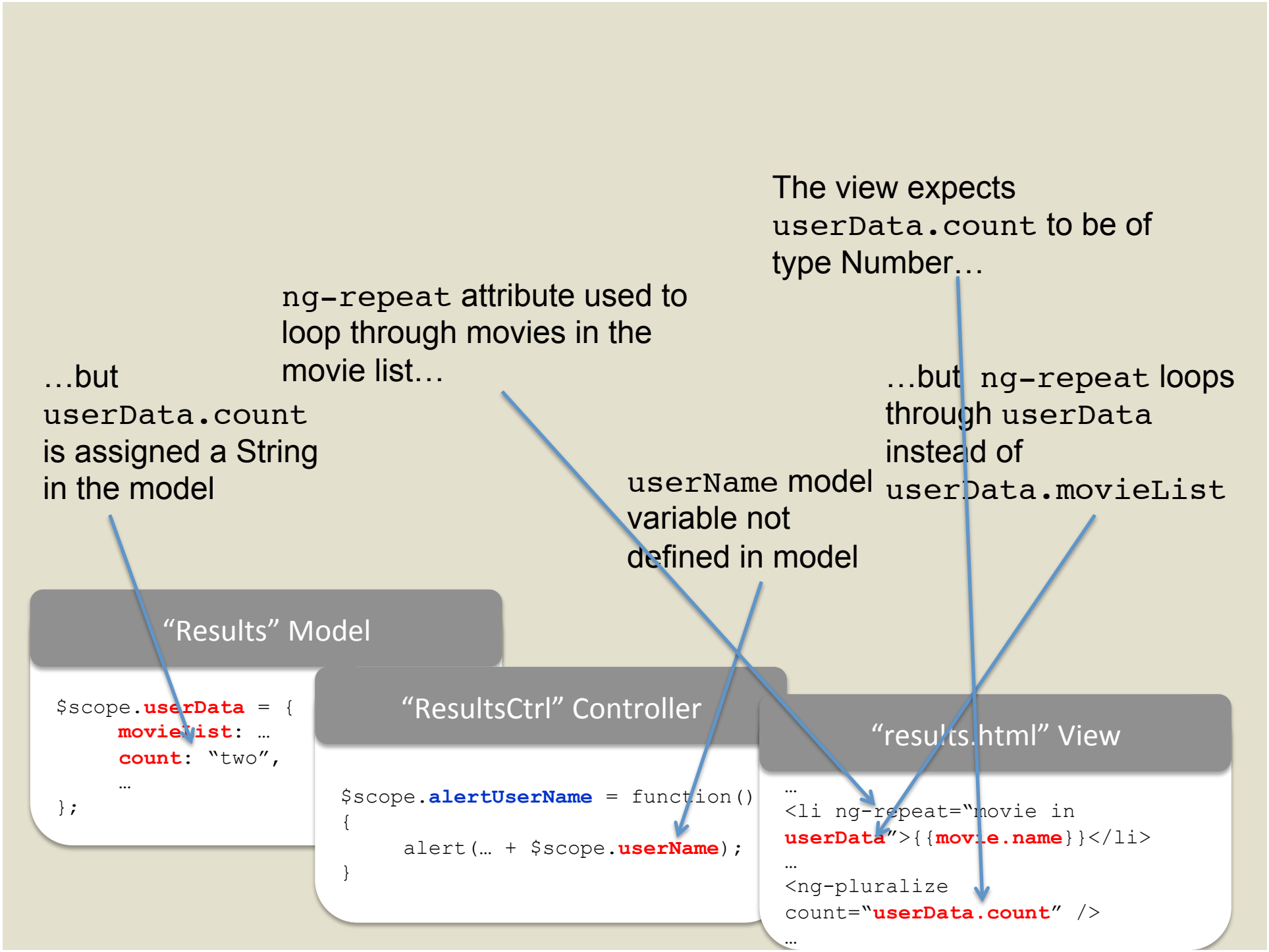
Groupings



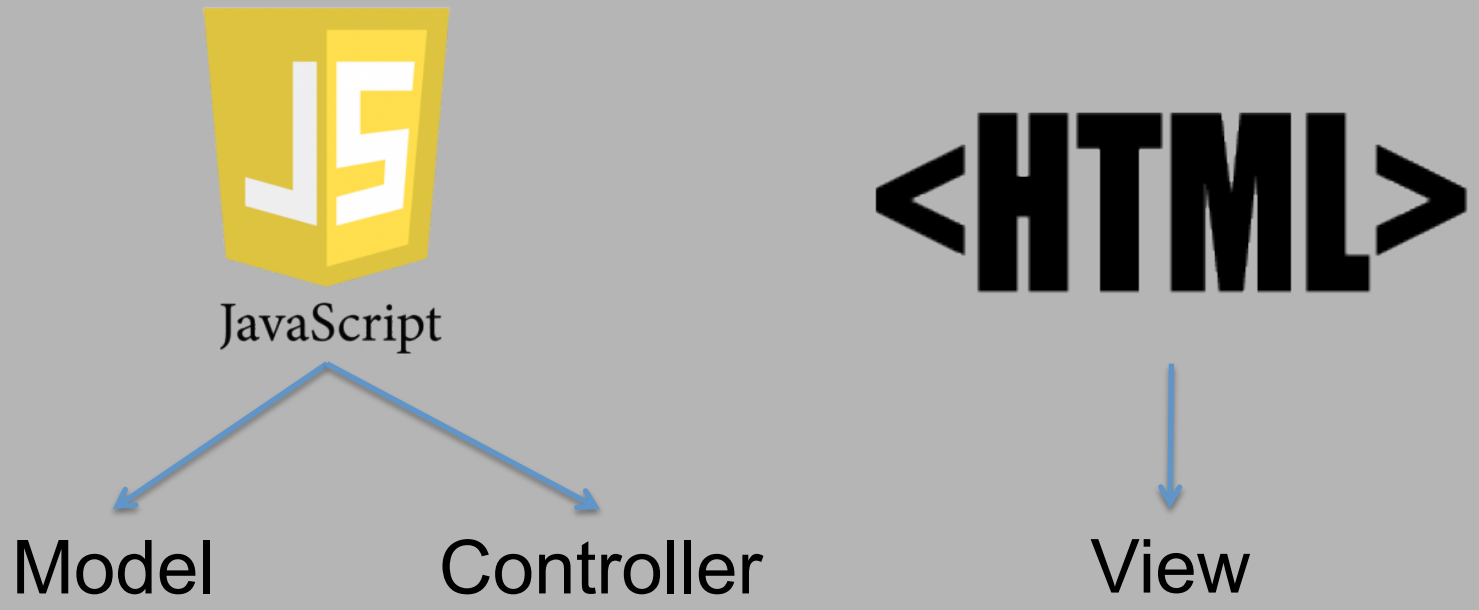
Running Example




“Search” Model
“Search” Grouping
“SearchCtrl” Controller
“search.html” View
“Results” Model
“Results” Grouping
“ResultsCtrl” Controller
“results.html” View


The view expects userData.count to be of type Number…
ng-repeat attribute used to loop through movies in the movie list…
…but userData.count is assigned a String in the model
userName model variable not defined in model
…but ng-repeat loops through userData instead of userData.movieList
"Results" Model
$scope.userData = {
movieList: …
count: "two",
…
};
"ResultsCtrl" Controller
$scope.alertUserName = function()
{
alert(… + $scope.userName);
}
"results.html" View
…
<li ng-repeat="movie in userData">{{movie.name}}
…
<ng-pluralize
count="userData.count" />
…

Finding the Models, Views, and Controllers



“Results” Model

“SearchCtrl” Controller

“ResultsCtrl” Controller

“search.html” View

“Search” Model

“results.html” View

Inferring Identifiers

Model Variable and Controller Function

Definitions

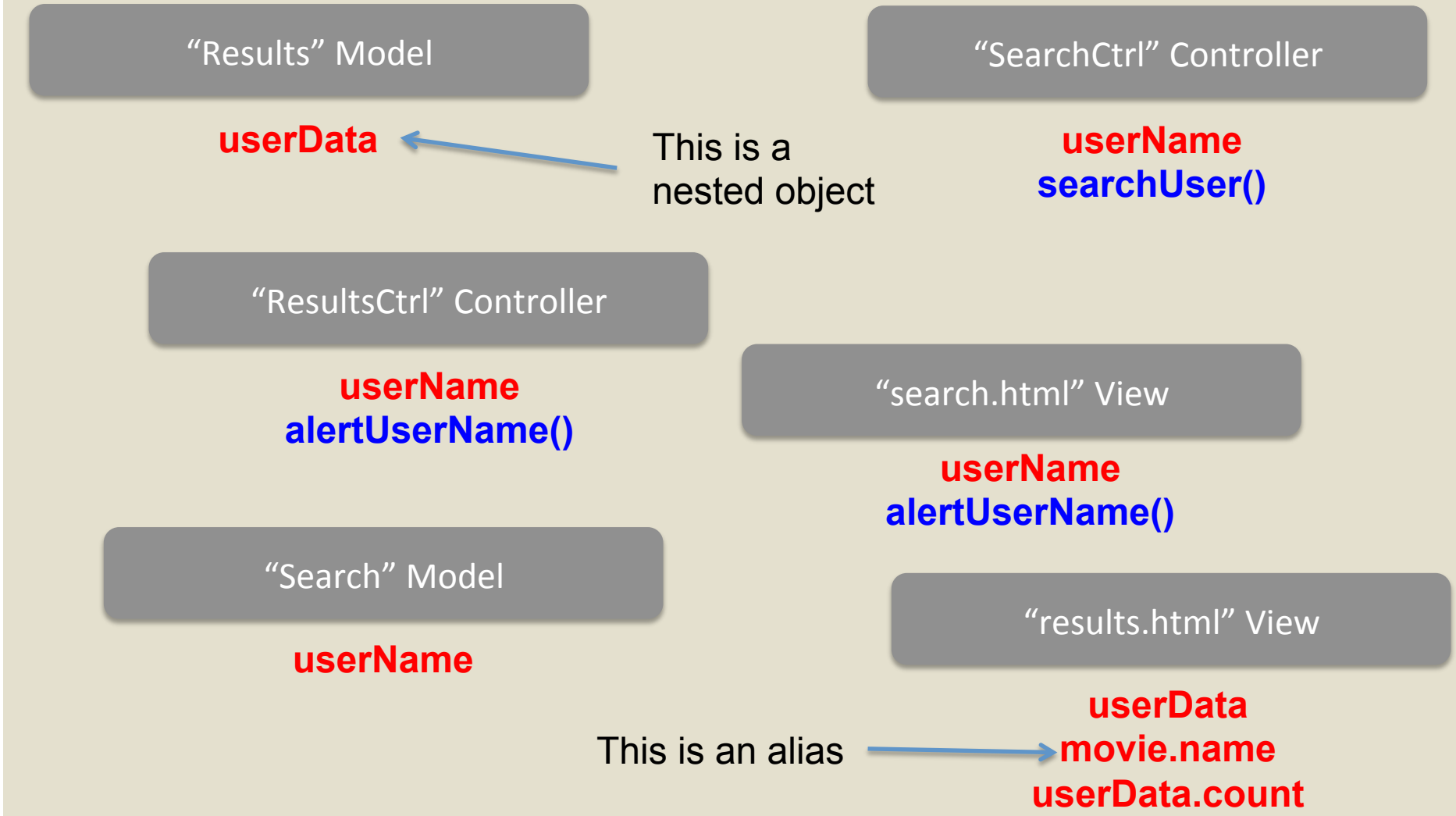
Done via `$scope` identifier

Model Variable and Controller Function

Usages

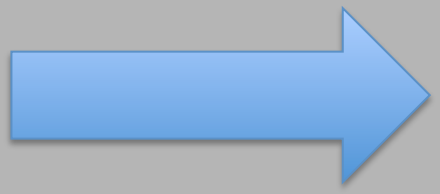
Embedded in HTML code

"Straw man" approach: Just take whatever identifiers you see in the above parts of the code


"Results" Model
userData
This is a
nested object
"SearchCtrl" Controller
userName
searchUser()
"ResultsCtrl" Controller
userName
alertUserName()
"search.html" View
userName
alertUserName()
"Search" Model
userName
"results.html" View
userData
movie.name
userData.count
This is an alias

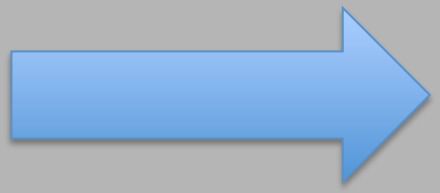
Inferring Identifiers: Nested Objects

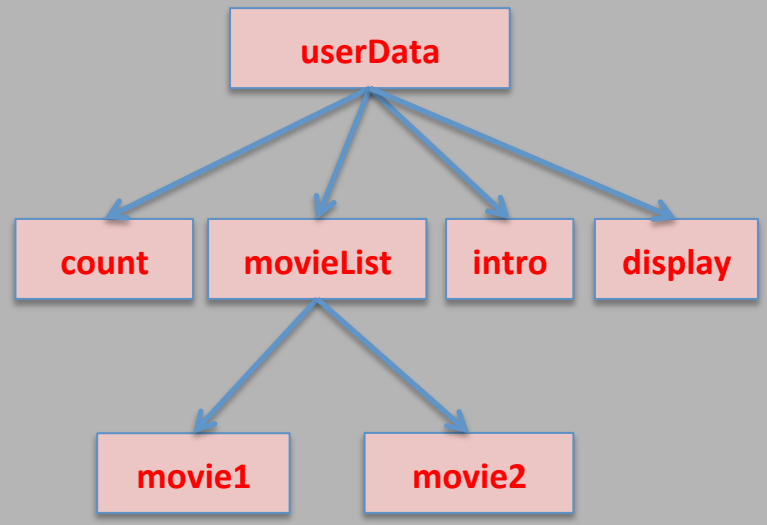
Object Literal



Tree, where each node is an object property identifier

```
$scope.userData = {
    movieList: {
        movie1: …,
        movie2: …
    },
    count: "two",
    intro: …,
    display: true
};
```





Inferring Types

For assigned/returned types: Look at AST node!

```
$scope.bar = "hello"
```

`bar` is of type String

StringLiteral

Insight: Parts of MVC applications typically written in "declarative" way

Inferring Types

For expected types: Examine HTML attribute

```
<ng-pluralize count="userData.count" />
```

`userData.count` is expected to be a Number

```
<h3 ng-if="userData.display">…</h3>
```

`userData.display` is expected to be a Boolean

"Results" Model

Object: userData
String: userData.count
Object: userData.movieList

"SearchCtrl" Controller

userName
Void: searchUser()

"ResultsCtrl" Controller

userName
Void: alertUserName()

"search.html" View

userName
alertUserName()

"Search" Model

String: userName

"results.html" View

Array/Object: userData
userData.count.name
userData.movieList.name
Number: userData.count

Discovering MVC Groupings

Groupings can be inferred via

`ng-controller`

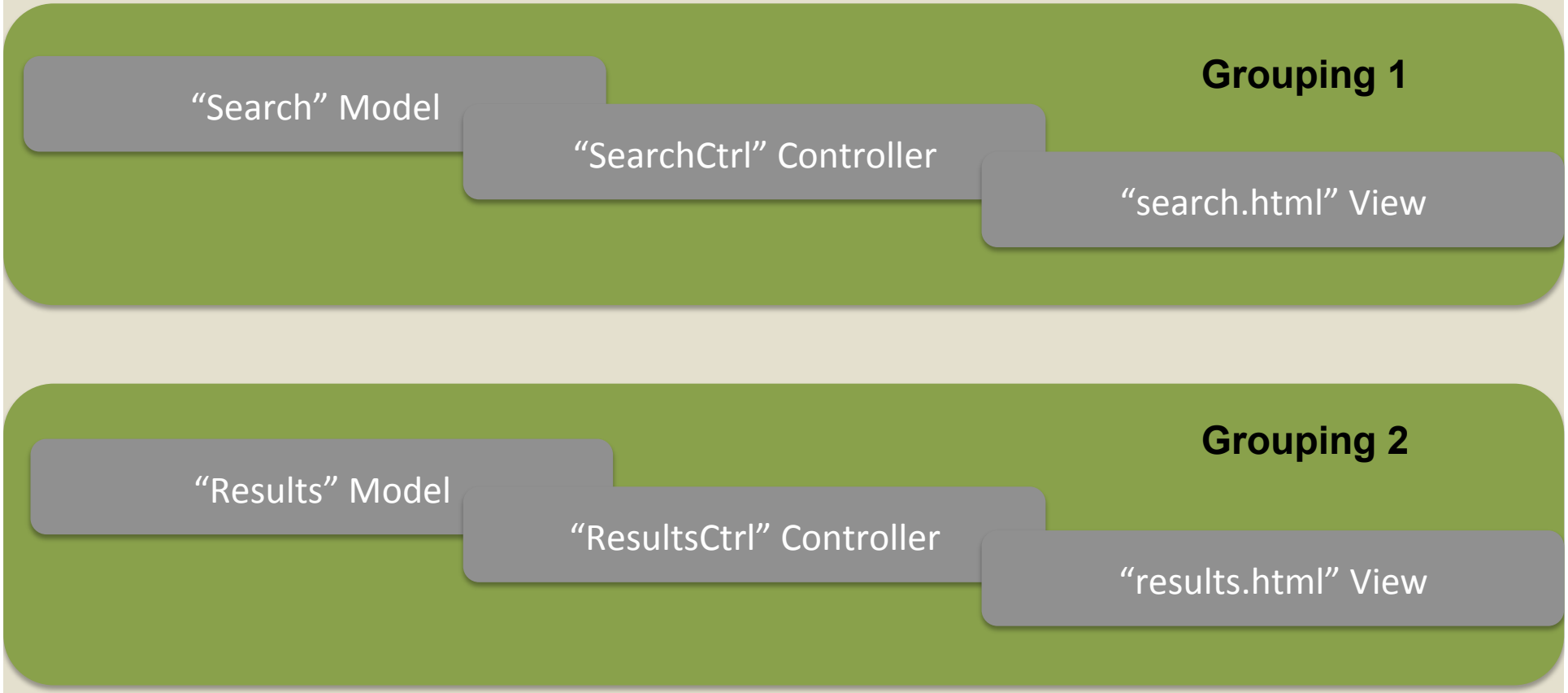
attribute

Groupings can be inferred via

routers

```
.when('/', {
    controller: 'SearchCtrl',
    templateUrl: 'search.html'
}
```

```
.when('/', {
    controller: 'ResultsCtrl',
    templateUrl: 'results.html'
}
```

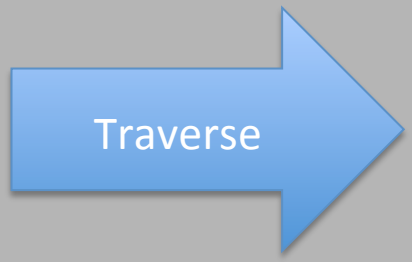

Grouping 1
“Search” Model
“SearchCtrl” Controller
“search.html” View
Grouping 2
“Results” Model
“ResultsCtrl” Controller
“results.html” View

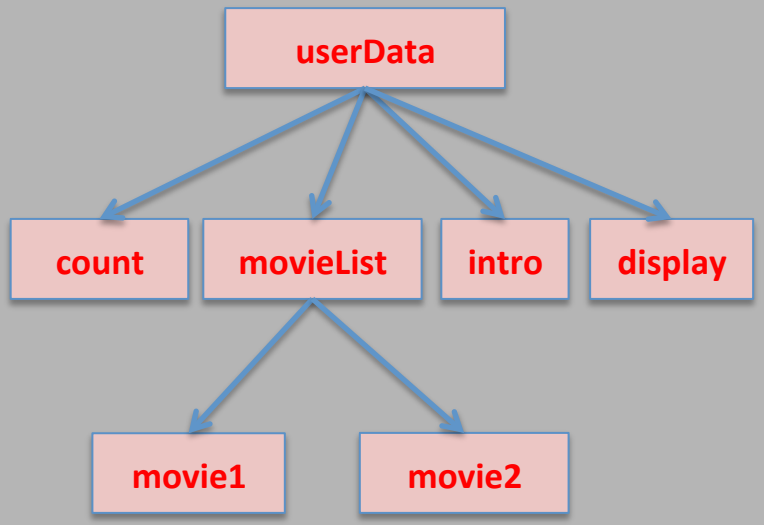
Detecting Inconsistencies

Controller Functions : String comparison of identifiers and types

Model Variables:

`userData.count`





Message 1:
`userName` is not defined in “Results” model

Message 2:
`userData.count.name` not defined in “Results” model

Message 3:
`userData.count` expected to be of type Number, but is of type String instead in “Results” model

Aurebesh

http://www.ece.ubc.ca/~frolino/projects/aurebesh

APP.JS

"count" is expected to be a Number!

```
29 searchApp.controller('ResultsCtrl', function($scope, $routeP
30     $scope.userData = {
31         movieList: getList($routeParams.userId),
32         intro: "Welcome User #" + $routeParams.userId,
33         display: true,
34         count: "two"
35     };
36     $scope.movieForms = {
37         one: '{} movie',
```

RESULTS.HTML

```
7  <div id="movieCount">
8      <ng-pluralize count="userData.count" when="movieForms"></n
9  </div>
10 <br />
11 <button ng-click="alertUserName()">
12     Which User?
13 </button>
```

Research Questions

RQ1: Can Aurebesh help developers find real bugs in real-world web applications?

RQ2: How accurate is Aurebesh in finding identifier and type inconsistencies?

RQ3: How quickly can Aurebesh perform the inconsistency detection analysis?

20 open-source AngularJS applications

100+ to **1000+** lines of JS code

eval() ✗

dynamic type conversions ✗

RQ1: Real Bugs

Aurebesh found ***15 bugs*** previously undetected (5 were acknowledged by developers)

Todo Application

Can’t add todo items…

Slide Maker

Can’t change themes and transitions in slides…

RQ2: Accuracy

Fault injection experiment

10 types of mutations

200 injections per application

Recall

$$\frac{\text{\# of successful detections}}{\text{\# of total injections}}$$

→ **96%** overall

Precision

$$\frac{\text{\# of successful detections}}{\text{\# of error messages displayed}}$$

→ Close to **100%** (only one false positive)

RQ3: Performance

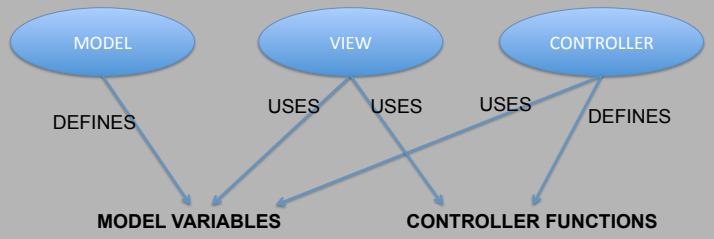
Average Execution Time

121 ms

Worst-case Execution Time

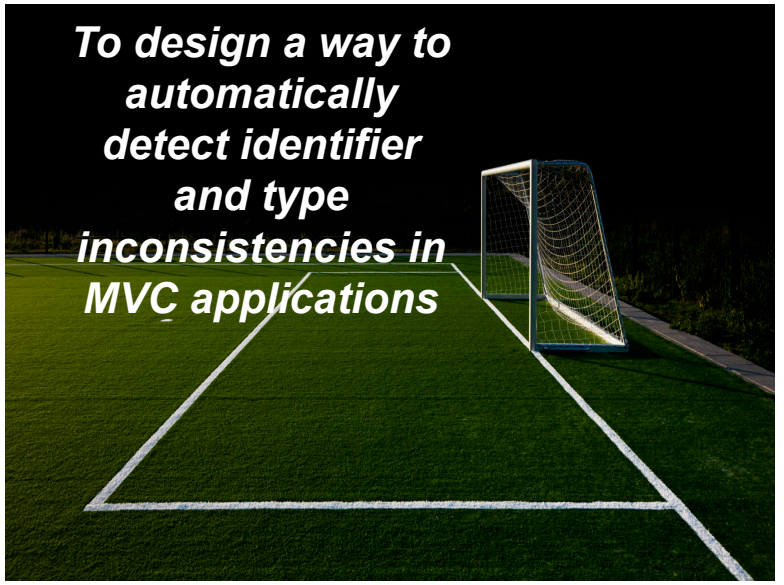
849 ms (eTuneBook)

MVC Frameworks to the Rescue!



*MVC Frameworks use **two-way data binding** ➔ No DOM method calls!*

4



KUMUNE

Aurebesh

http://www.ece.ubc.ca/~frolino/projects/aurebesh

APP.JS

"count" is expected to be a Number!

```
29  searchApp.controller('ResultsCtrl', function($scope, $routeP
30      $scope.userData = {
31          movieList: getList($routeParams.userId),
32          intro: "Welcome User #" + $routeParams.userId,
33          display: true,
34          count: "two"
35      };
36      $scope.movieForms = {
37          one: '{} movie',
```

RESULTS.HTML

```
7   <div id="movieCount">
8       <ng-pluralize count="userData.count" when="movieForms"></n
9   </div>
10  <br />
11  <button ng-click="alertUserName()">
12      Which User?
13  </button>
```

29